Epidemic Algorithms For Replicated Database Maintainance

Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry

University of Maryland, Baltimore County
CMSC 621: Advanced Operating System
Presented by: Vivek Relan (relan1 AT umbc DOT edu)
Table of Content

- Motivation
- Model of epidemics
- Epidemic methods
  - Anti-entropy
  - Rumor mongering
- Death certificate
- Summary
Basic goal/ Motivation

- Distribute some data among a group of nodes
  - Should be fast, but no synchrony guarantees
  - Should be robust (some nodes may crash, but still works)
  - Should scale to many nodes
  - Should be efficient
Model of epidemics

- **Epidemics** study the spread of disease or infection in terms of population of infected/uninfected individuals and their rate of change.

- **Susceptible**
  If node has not yet received an update.

- **Infective**
  If node holds an update and it is willing to share.

- **Removed**
  If node has the update but no longer willing to share.
Model of epidemics (cont.)

- **How does it work?**
  - Initially, a single individual is infective
  - Individuals get in touch with each other, spreading the update

- **Rumor spreading**, or gossiping, is based on the same principles

- **Can we apply same ideas to distributed systems?**
  - Goal is to spread the infection (update) as fast as possible!
The goal

- When database is replicated at many sites, maintaining consistency in the presence of updates is a significant problem.

- The goal of update distribution process is to drive the system towards consistency

  For all $S, s' \in S$: $s.value = s'.value$

- And we want to achieve this with algorithms that are efficient, robust and scalable.
Epidemic Methods

- **Direct mail/ Best effort delivery**
  Each host sends all updates to every other host

- **Anti-entropy**
  Sites periodically contact other sites and reconcile database with them

- **Rumor mongering**
  When a site encounters a new update, it begins to gossip it to random sites until the rumor becomes old by some measurement
Anti-entropy

- Node s executes:
  
  repeat periodically, every delta time units
  r = selectPeer(S – {s})
  resolveDifference(s, r)

- Push
  
  if (s.value.time > r.value.time)
      r.value = s.value

- Pull
  
  if (s.value.time < r.value.time)
      s.value = r.value

- Push-pull
Anti-entropy: Convergence

- To analyze convergence, we must consider what happens when only few nodes remain susceptible.

- Both starts to converge to 0, but pull is more rapid, so in practice pull is used.

![Graph showing the convergence of susceptible nodes over cycles.](image-url)
Anti-entropy: Comments

**Benefits**
- Simple epidemics eventually infect all the population
- For a push implementation, the expected time to infect everyone is $\log(n) + \ln(n)$

**Drawbacks**
- Propagates updates much slower than direct mail
- Requires examining contents of database even when most data agrees, so it can not practically be used too often
Rumor mongering

- Nodes are initially ignorant or susceptible

- When a node receives a new update it becomes a hot rumor and infective

- A node that has a rumor periodically chooses randomly another node to spread the rumor

- Eventually, a node will lose interest in spreading the rumor and becomes removed
  - Spread too many times
  - Everybody knows it

- A sender can hold and transmit a list of infective updates rather than just one
Rumor mongering: loss of interest

- Counter v/s coin
  - Coin (random): lose interest with probability $1/k$
  - Counter: lose interest after $k$ contacts

- Feedback v/s blind
  - Feedback: lose interest only if the recipient knows the rumor
  - Blind: lose interest regardless of the recipient

- How fast does the system converge to a state where all nodes are not infective?
  - Eventually, everybody will lose interest

- Rumor may stop before reaching all nodes
Rumor mongering analysis

- For Feedback and coin
  - Let s, i, and r denote fraction of susceptible, infective, and removed nodes respectively.

\[
\begin{align*}
  s + i + r &= 1 \\
  \frac{ds}{dt} &= -si \quad \text{(Feedback)} \\
  \frac{di}{dt} &= +si - \left(\frac{1}{k}\right)(1-s)i \quad \text{(Coin)}
\end{align*}
\]

- Solving the equation: \( s = e^{-(1+k)(1-s)} \)
  - Thus, increasing \( k \) we can make sure that most nodes get the rumor, exponentially better
Quality measures

- **Residue**
  - The nodes that remain susceptible when the epidemic ends: value of $s$ when $i=0$
  - Residue must be as small as possible

- **Traffic**
  - The average number of database updates sent between nodes
  - $m = \text{total update traffic} / \# \text{ of nodes}$

- **Delay**
  - $t(\text{avg})$: average time it takes for the introduction of an update to reach a node
  - $t(\text{last})$: time it takes for the last node to get the update
Deletion and death certificate

- **Deletion**
  - We can not delete an entry just by removing it from a node – the absence of the entry is not propagated.
  - If the entry has been updated recently, there may still be an update traversing the network.

- **Death certificate**
  - Solution: replace the deleted item with a death certificate that has a timestamp and spreads like an ordinary update.
Deletion and death certificate

- **Problem**
  - We must at some point, delete death certificates or they may consume significant space.

- **Strategy I**
  - Retain each DC until all nodes have received it. Requires a protocol to determine which nodes have it and to handle node failures.

- **Strategy II**
  - Hold DC for some time and discard them. Pragmatic approach, still have the “resurrection” problem; increasing the time requires more space.
Dormant certificate

- If a DC is older than the time it takes to propagate it to all nodes, it is highly unlikely anyone still has an old copy.

- So, we can delete very old DC but retain only a few “dormant” copies in some nodes

- If an obsolete update reaches a dormant DC, it is “awakened” and re-propagated.
Summary

- Presents randomized, epidemic algorithm for distributing updates in a replicated database to approach consistency

- Analyzes performance of two random epidemic algorithms – anti-entropy, and rumor mongering

- Emphasize importance of spatial distribution for efficiency